VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Využívání IoT při detekci správného používání zdravotních pomůcek

Use of IoT in Detecting Proper use of Medical Aids

2019 David Bulawa

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **David Bulawa**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Využívání IoT při detekci správného používání zdravotních pomůcek
Use of IoT in Detecting Proper use of Medical Aids

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je implementovat program pro dětskou helmičku, která se používá v případech kdy je u dětí zjištěn nesprávný růst lebečních kostí. U takto používané helmičky je nutné její pravidelné používání. Práce si klade za cíl vytvořit program pro dodaný hardware, který bude pomocí indukčního čidla měřit vzdálenost hlavy od helmičky a tak bude detekovat její správné používání. Měřená data budou zasílána do sítě IoT Sigfox.

1. Popište všechny podstatné komponenty dodaného hardware a stručně popište síť IoT Sigfox.
2. Zprovozněte základní komunikaci přes síť Sigfox.
3. Vytvořte program, kterým bude možné vyčítat hodnoty z čidla indukčnosti. Měřené hodnoty převeďte na vzdálenost. Naměřená data zasílejte do sítě Sigfox.
4. Optimalizujte program tak, aby bylo dosaženo co nejmenší spotřeby, která umožní provoz v řádu měsíců.
5. Realizujte jednoduchou webovou aplikaci pro vyčítání dat zaslaných do sítě Sigfox.
6. Program řádně zdokumentujte.

Seznam doporučené odborné literatury:

[1] BARR, Michael. Programming embedded systems in C and C++. Sebastopol, Calif.: O'Reilly, c1999. ISBN 1565923545.
[2] LPC82x - NXP semiconductors https://www.nxp.com/docs/en/data-sheet/LPC82X.pdf
[3] AX-SIGFOX-API SIGFOX Compliant Software Stack
http://www.onsemi.com/pub/Collateral/AND9478-D.PDF

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Seidl, Ph.D.**

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *30. dubna 2019*

.....
podpis studenta

Rád bych poděkoval Ing. Davidu Seidlovi, Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce.

Abstrakt

Cílem této práce je vytvoření aplikace kontrolující pokrok a správné používání pro helmu na tvarování hlavičky u dětí velmi nízkého věku. Tato helma je používána u kojenců trpících plagiocefálií (asymetrickým růstem lebky).

Pro řešení problémů byly využity následující kompomenty: mikropočítač, mobilní síť Sigfox a inerciální senzor pro měření vzdálenosti. Návrh celého systému musel zohledňovat nutnost dlouhodobého provozu na baterie, bylo tedy nutné dbát na nízkou spotřebu, jelikož aplikace musí být schopna fungovat několik měsíců bez nutnosti měnit baterii.

Vytvořené řešení pomocí inerciálního senzoru kontroluje správnost nošení helmy a měří přesnou vzdálenost senzoru umístěného na helmě od lebky. Zároveň bylo dosaženo nízké spotřeby díky použití vhodných součástek.

Hlavním výsledkem této práce je program pro mikroprocesor, který automaticky kontroluje správné nošení helmy. Po zadání povelu uživatelem provede program detailní měření a odešle data na server prostřednictvím sítě Sigfox. Očekávaná doba nošení této helmy je 4 měsíce a po tuto dobu není potřeba měnit baterie.

Klíčová slova: Arduino, Mikropočítač, Internet věcí, Sigfox, plagiocefálie, SPI, USB

Abstract

The goal of this thesis is to create an application able of controlling progress and proper use of skull shaping helmet for children of very young age. This helmet is used for babies diagnosed with plagiocephaly (assymetrical growth of skull).

Components used for solving this problem were: microcontroller, mobile network Sigfox and inductance sensor. Design of entire system had to account for long-term service using only small battery. Therefore it was necessary to keep the power consumption as low as possible. Application is required to run for several months without need to change the battery.

Solution is using inductance sensor to ensure proper use of helmet and measures precise distance between the sensor on helmet and childs skull. Low power consumption has been accomplished by using proper components.

Main result of this thesis is program for microntroller which automatically checks proper wearing of helmet. Program will also execute detailed measurement and send data to server via Sigfox network after receiving user input. Expected time of wearing this helmet is 4 months and it isnt necessary to change batteries during this time period.

Key Words: Arduino, Microcontroller, Internet of Things, Sigfox, plagiocephaly, SPI, USB

Obsah

Seznam použitých zkratek a symbolů

IoT	–	Internet of Things
WFI	–	Wait for Interrupt
RTC	–	Real-time counter
SPI	–	Serial Peripheral Interface
USB	–	Universal Serial Bus
UART	–	Universal asynchronous receiver-transmitter
MCU	–	Microcontroller unit
IRC	–	Internal reference clock
PMU	–	Power management unit
AHB	–	Advanced High-performance Bus
APB	–	Advanced Peripheral Bus
MSB	–	Most significant bit
LSB	–	Least significant bit

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

U dětí ve věku několika měsíců se může začít projevovat jistá deformovanost lebky. Jednou z nejčastějších příčin je plagiocefálie. Tento stav se léčí pomocí speciální helmy, která slouží jako forma, do které má lidská lebka dorůst. Děti většinou potřebují tuto helmu nosit po dobu tří až šesti měsíců. Se stále rostoucím technologickým pokrokem dochází k vývoji i v tomto odvětví. Instalací malého obvodu do helmy se výrobce může ujistit, že dítě skutečně nosí helmu tak často, jak je potřeba pro požadované výsledky. Doposud se pro kontrolu používalo bluetooth zařízení, které komunikovalo s mobilním telefonem rodičů.

Tato práce se zabývá aplikací malého obvodu, který nahradí bluetooth za IoT vysílač a tím odstraní potřebu propojení s dalším zařízením. Hlavním účelem obvodu je samotné měření růstu lebky. Toto měření probíhá pomocí digitálního senzoru vzdálenosti. Tento senzor je také použit ke kontrole správnosti nošení helmy, kterou je potřeba nosit alespoň 22 hodin denně, aby došlo k žádaným výsledkům. Všechny naměřené údaje budou k dispozici výrobcům a zdravotnímu personálu.

K přenosu měřených dat bude použita technologie IoT, která je vytvořena pro přenos malého počtu dat. Dále se v práci budu zabývat spotřebou celého obvodu, kterou je potřeba držet na co nejnižší úrovni.

2 Použité komponenty

Hlavním cílem projektu je vytvořit modul schopný měřit vzdálenost do několika milimetrů při zachování přesnosti v řádech setin milimetrů a současně komunikovat po síti IoT Sigfox. K ovládání celého modulu slouží mikroprocesor, který komunikuje s měřícím senzorem i s rozhraním pro komunikaci po síti IoT. Zapojení obvodu je znázorněno v obr. 1. Všechny součástky byly vybírány s ohledem na nízkou spotřebu.

V průběhu projektu bylo postupně využito více procesorů. K použití celkem třech procesorů došlo s postupně se měnícími potřebami projektu a zjišťováním nedostatků předchozích procesorů.

2.1 Procesor NCS36510 [1]

Tento procesor je dodáván firmou ON Semiconductor mimo jiné jako součást IoT Development kitu[7] pro Sigfox. Procesor patří do rodiny ARM Cortex-M3 a je navržen pro aplikace vyžadující minimální spotřebu, umožňuje použití Coma sleep módu, při kterém se spotřeba pohybuje v jednotkách uA. Procesor nabízí 640kB FLASH paměti pro samotný program a 48kB RAM paměti pro uchovávání dat.

Procesor NCS36510 byl zvolen pro vývoj jako první, protože byl dodáván jako součást IoT development kitu[7] od výrobce On Semiconductor a současně splňoval všechny nároky na něj kladené.

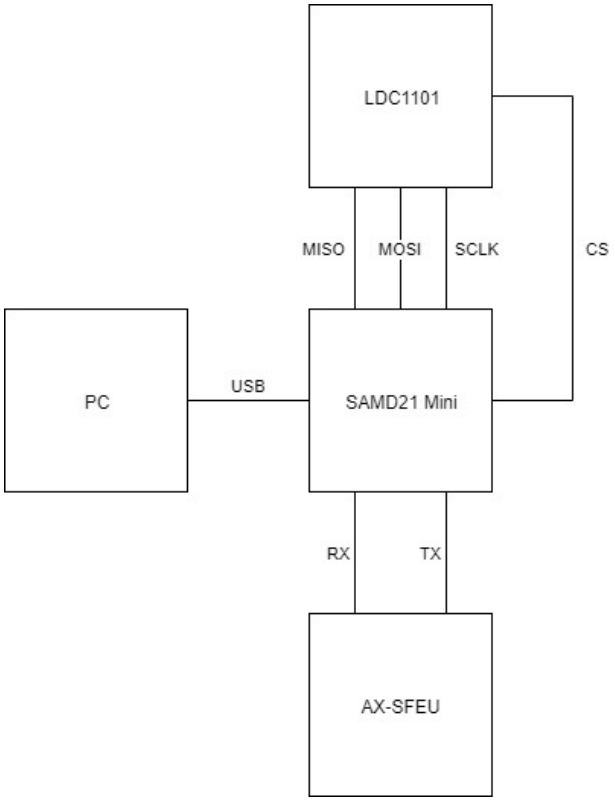
Základní vlastnosti procesoru: [1]

- jádro 32-bit ARM Cortex M3
- 640KB flash paměti, 48KB SDRAM paměti
- spotřeba v run módu 16mA (Závislé na aktuální konfiguraci procesoru)
- spotřeba v Coma sleep módu: 0.65uA
- provozní napětí 1.0V až 3.6V

2.1.1 Režimy spotřeby

2.1.1.1 Run V tomto módu jsou všechny digitální systémy procesoru spuštěny a napájeny a procesor vykonává program.

2.1.1.2 Sleep V tomto módu jsou napájeny všechny digitální systémy. Spuštěny jsou všechny systémy s výjimkou generátoru hodinového signálu, jehož vypnutím je způsobeno zastavení vykonávání programu. Při zjištění přerušení procesor přejde do Run módu a provádí program začínaje posledním známým stavem.



Obrázek 1: Blokové schéma testovacího obvodu.

2.1.1.3 Deep Sleep Tento mód je podobný Sleep módu. Rozdíl je, že tento mód nenapájí flash paměti. Procesor nevykonává program, jelikož je jeho generátor hodinového signálu vypnutý. Při detekci přerušení jsou nejprve napájeny flash paměti a následně procesor přejde do Run módu.

2.1.1.4 Coma V tomto módu jsou v digitálním systému napájeny pouze paměťové registry. Všechny registry si tedy uchovávají své hodnoty. Dochází k vypnutí rychlých oscilátorů. Pomalé oscilátory slouží jako generátor hodinového signálu pro časovač přerušení. Při detekci přerušení se začne napájet flash paměť a zbytek digitálních systémů. Procesor následně přejde do Run módu a začne vykonávat program za místem, kde vstoupil do tohoto módu.

2.1.2 Programování

Pro tento procesor vydala firma onsemi vlastní vývojové prostředí zvané IDK [4], které je založené na velmi rozšířeném prostředí eclipse. Pro procesor jsou rovněž k dispozici knihovny pro práci se zamýšleným modulem pro komunikaci po síti Sigfox. Tento procesor se poté programuje v jazyce C nebo C++, podle volby programátora. Procesor je podporován projektem mbed.

Od tohoto obvodu bylo potřeba odstoupit, protože v době řešení projektu bylo velmi složité tento obvod zakoupit v malých množstvích. Nejnižší objednávací množství bylo dva tisíce kusů, což je pro vývoj nepřípustné.

2.2 Procesor LPC824 [2]

Druhým modelem procesoru, na kterém probíhal vývoj byl procesor LPC824 od firmy NXP patřící do rodiny ARM Cortex-M0+. Tento procesor nabízí usporný režim zvaný Power-down mode podobný Coma sleep módu u procesoru NCS36510. Spotřeba v tomto módu by neměla převýšit 10uA. Tento procesor má 32kB flash paměti pro program a 8kB flash paměti pro data.

Základní vlastnosti procesoru:

- jádro 32-bit ARM Cortex M0+
- 32KB flash paměti, 8KB SDRAM paměti
- spotřeba v run módu 20uA (Závislé na aktuální konfiguraci procesoru)
- spotřeba v Deep power-down módu: 1uA
- provozní napětí 1.8V až 3.6V

2.2.1 Režimy spotřeby

2.2.1.1 Active V tomto módu jsou všechny digitální systémy procesoru napájeny a spuštěny a procesor vykonává program.

2.2.1.2 Sleep Při aktivaci Sleep módu je zastaven generátor hodinového signálu pro procesor. Probuzení z tohoto režimu vyžaduje pouze aktivaci generátoru hodinového signálu. V tomto módu je provádění programu zastaveno dokud nenastane přerušení nebo reset. Periférní funkce v tomto módu stále operují a mohou vygenerovat přerušení pro probuzení procesoru z tohoto módu.

2.2.1.3 Deep-sleep V tomto módu je procesor uveden do Sleep módu popsaného výše a všechny hodiny vyjma IRC(Internal reference clock) a watchdog oscilátoru jsou vypnuté. Všechny analogové bloky jsou vypnuty a flash paměť je ve standby módu. V tomto módu může program nastavit watchdog timer na probuzení dle vlastního načasování. Z tohoto módu lze procesor probudit pomocí resetu, pomocí digitálních pinů zvolených jako vstupy pro přerušení, pomocí časovače watchdog nebo pomocí přerušení přes rozhraní USART(pokud je nakonfigurován v sychnronním slave módu), SPI a I2C(Ve slave módu).

Jakékoliv přerušení použité pro přechod z tohoto režimu musí být povoleno v jednom z SYSCON registrů pro umožnění buzení.

2.2.1.4 Power-Down V Power-down módu je procesor ve Sleep módu a všechny hodinové zdroje vyjma watchdog jsou vypnuty. Navíc jsou vypnuty všechny analogové bloky a flash paměti.

Power-down mód může být ukončen pomocí resetu, pomocí digitálních pinů zvolených jako vstupy pro přerušení, pomocí časovače watchdog nebo pomocí přerušení přes rozhraní USART(pokud je nakonfigurován v sychnronním slave módu), SPI a I2C(Ve slave módu).

Jakékoliv přerušení použité pro buzení z tohoto režimu musí být povoleno v jednom z SYSCON registrů pro umožnění buzení.

Tento režim je v porovnání s Deep-sleep módem úspornější na spotřebu energie, ale Deep-sleep mód nabízí rychlejší přechod do aktivního režimu.

2.2.1.5 Deep power-down V režimu Deep power-down je napájení vypnuto pro celý obvod vyjma WAKEUP pinu a časovače probuzení pokud je umožněn. Pro uchovávání dat v tomto módu jsou k dispozici 4 všeobecné registry.

Z tohoto režimu může být procesor buzen přes externí signál pomocí WAKEUP pinu nebo bez externího signálu pomocí time-out od self-wake-up timeru.

2.2.2 Programování

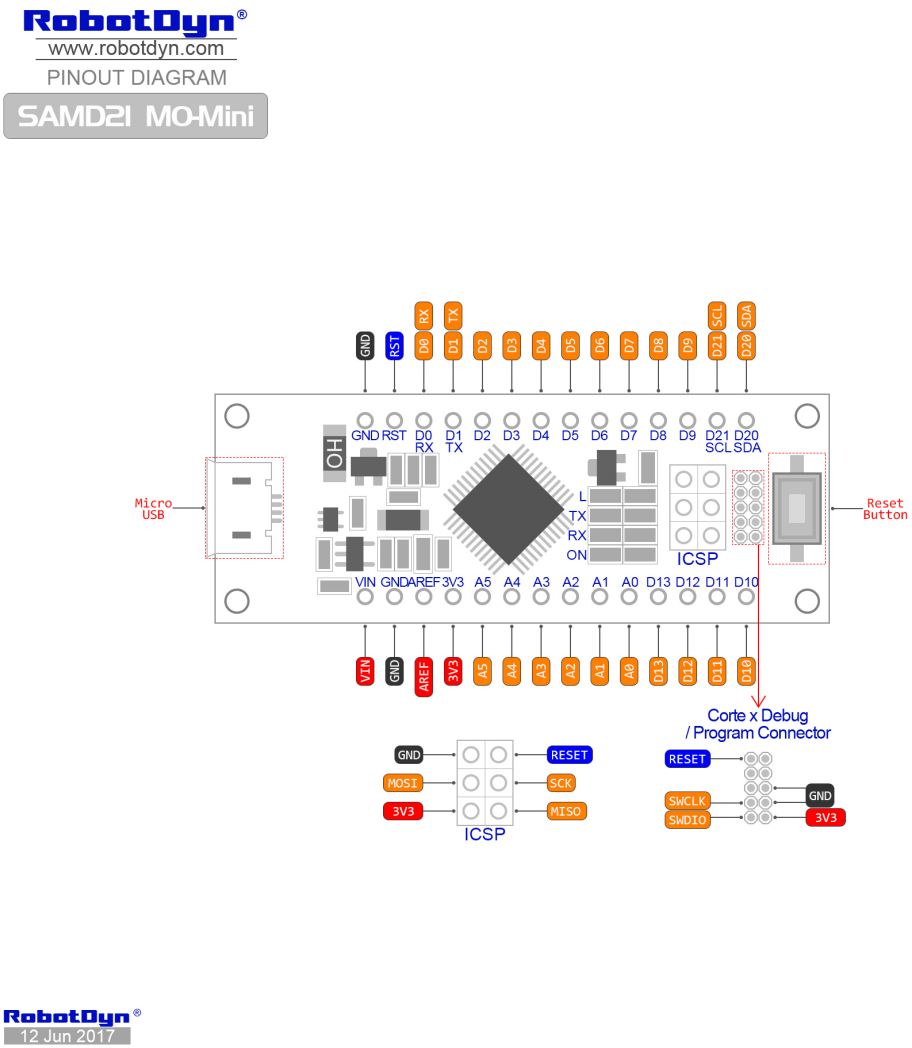
Pro programování tohoto procesoru bylo využito IDE MCUXpresso[5], které sjednocuje práci na všech ARM procesorech vyrobených firmou NXP. Toto IDE je nádstavbou rozšířeného prostředí eclipse. Programovací jazyk pro procesor je stejně jako u předchozího modelu C nebo C++ dle volby programátora, procesor je také podporován projektem MBED.

2.3 Procesor ATSAMD21G18 [3]

Finální procesor použitý pro vývoj je ATSAMD21G18 od výrobce Atmel. Tento procesor je postavený na 32 bitové architektuře Arm Cortex M0+ a běží na frekvenci do 48MHz. Procesor nabízí 4 různé úsporné režimy - 3 konfigurace IDLE režimu a Standby režim. Procesor má 256KB flash paměť a 32KB SDRAM paměť.

Základní vlastnosti procesoru: [3]

- jádro 32-bit ARM Cortex M0+
- 256KB flash paměti, 32KB SDRAM paměti
- spotřeba v run módu 30mA
- spotřeba v deep sleep módu: 20uA
- provozní napětí 1.62V až 3.63V



Obrázek 2: SAMD21mini pinout `https://robotdyn.com/samd21-m0-mini.html`

2.3.1 Vývojový kit SAMD21 M0-Mini

Vývoj programu probíhal na vývojové desce SAMD21 mini od výrobce robotdyn.Tato deska je 32 bitovou nádstavbou platformy Arduino UNO fungující právě na procesoru ATSAMD21G18 (obr. 2).

2.3.2 Režimy spotřeby

Tabulka 1: Přehled úsporných režimů procesoru SAMD21

Sleep Mode	CPU Clock	AHB Clock	APB Clock	Main Clock	Režim regulátoru	Režim RAM
Idle0	Stop	Run	Run	Run	Normal	Normal
Idle1	Stop	Stop	Run	Run	Normal	Normal
Idle2	Stop	Stop	Stop	Run	Normal	Normal
Standby	Stop	Stop	Stop	Stop	Low power	Low power

2.3.2.1 Active V tomto módu jsou všechny systémy procesoru spuštěny a napájeny. Probíhá vykonávání programu.

2.3.2.2 IDLE Procesor nabízí 3 různé úrovně IDLE režimů. U všech dochází k zastavení procesoru. Uživatel může měnit úroveň IDLE režimu vypnutím časovacích modulů a zdrojů konfigurací skupiny bitů SLEEP.IDLE.

2.3.2.3 Standby Standby mód umožňuje dosažení velmi nízké spotřeby. V tomto módu jsou zastaveny všechny časovače s výjimkou těch, které mají ONDEMAND bit nastavené na 0. Například časovač hodin reálného času RTC tak může v standby módu fungovat. V tomto případě bude zapnutý také hodinový generátor reálného času. Napěťový regulátor(Obvod starající se o správnou distribuci napětí.) a RAM operují v úsporném režimu.

2.3.3 Programování

Pro programování tohoto procesoru jsem použil platformu Arduino[6], která nabízí vlastní IDE. Programovací jazyk Arduino je nádstavbou jazyka C++, která má zabudované funkce potřebné pro práci s mikroprocesorem.

2.4 Senzor LDC1101 [9]

LDC1101 od společnosti Texas Instruments je převodník indukčnosti na digitální hodnoty pro snímání pozice, rotace nebo rychlého pohybu na krátké vzdálenosti. Umožňuje spolehlivé a přesné měření. Díky dvojímu měřícímu jádru umožňuje současně 2 různá měření o rozlišení 16 bitů a jedno měření s přesností 24 bitů. LDC1101 dále umožňuje přímé porovnání měřené hodnoty s programátorem nastavenou hodnotou, které může být dynamicky měněno za běhu zařízení. Díky tomu může obvod LDC1101 fungovat jako prostý spínač. Indukční měření umožňuje přesné měření lineární či úhlové pozice, pohybu, stlačení či vibrací. Všechny tyto výhody jsou k dispozici v obvodu o velikosti 3*3mm o 10 pinech. Komunikace s obvodem probíhá pomocí sběrnice SPI a lze ji provádět pomocí mikropočítače. Obvod lze také obsluhovat z počítače pomocí grafické aplikace LDC1101 EVM GUI dostupné pro operační systém Windows.

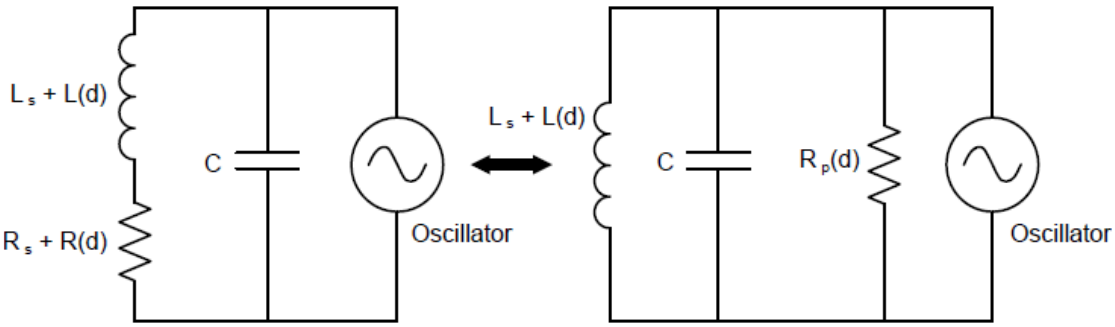
Základní vlastnosti obvodu[9]:

- provozní napětí 1.8V až 3V
- frekvence čidla: 500kHz - 10MHz
- spotřeba v Active módu: 1.9 mA
- spotřeba v Shutdown módu: 1.4 uA

2.4.1 Princip měření

Základní princip indukčního měření lineární pozice je založen na principu vířivého proudu. Střídavý proud při průchodu cívkou generuje střídavé magnetické pole. Přivedením vodivého materiálu do blízkosti cívky dojde k přenosu energie z oscilujícího magnetického pole na vodivý materiál. Takový přenos indukuje malé elektrické proudy na povrchu přiblíženého materiálu, právě tyto proudy jsou nazývány vířivými proudy. U těchto proudů vlivem odporu při cirkulaci vzniká malý úbytek energie, který se projevuje zahříváním. Těmto úbytkům se říká ztráty způsobené vířivými proudy. Tyto ztráty jsou závislé na vzdálenosti, velikosti, složení a orientaci vodivého materiálu k magnetickému poli. Indukované vířivé proudy na vodivém materiálu poté vytvářejí vlastní magnetické pole, které působí na původní pole vytvořené cívkou, tato reakce pozmění vlastnosti cívky.

Použije-li se ke generování střídavého magnetického pole cívka, důsledkem vysoké impedance se výrazně zvyšuje spotřeba elektrické energie. Tuto spotřebu lze značně zredukovat přidáním paralélního kondenzátoru. Touto modifikací se z obvodu stane LC rezonátor (obr. 3). Díky změně obvodu v LC rezonátor je spotřeba energie omezena pouze na úbytky na cívce a ztráty způsobené vířivými proudy. Obvod LDC1101 neměří přímo sériový odpor, místo toho měří paralélní



Obrázek 3: Sériové a paralélní zapojení LC obvodu propojeného s oscilátorem

rezonanční impedanci R_P. Pří meření R_P tedy probíhá pouze měření ztrát způsobených vířivými proudy.

$$R_P(d) = \left(\frac{1}{R_S + R(d)}\right)\left[\frac{L_S + L(d)}{C}\right] \tag{1}$$

Měření je prováděno monitorováním ztráty energie na rezonátoru při regulaci oscilační amplitudy na konstantní úroveň. LDC1101 může určit hodnotu R_P monitorováním množství energie dodané rezonátoru.

LDC1101 má pro měření impedance a rezonanční frekvence připojeného senzoru k dispozici dva na sobě nezávislé měřící bloky. R_P + L blok umožňuje současné měření impedance a rezonanční frekvence LC rezonátoru s rozlišením 16b. Pro měření ve vyšším rozlišení je k dispozici LHR blok, který měří pouze rezonanční frekvenci senzoru s rozlišením do 24b. K určení pozice cíle od cívky senzoru lze použít oba dva měřící režimy. Výsledky jsou poté zasílány po SPI připojenému MCU.

2.4.2 Režimy spotřeby

Kromě normálního aktivního režimu nabízí LDC1101 dva režimy pro omezení spotřeby. Ve Sleep módu si obvod zachovává hodnoty v registrech a může rychle přejít do aktivního módu a začít měřit. V Shutdown módu je spotřeba energie značně nižší, ale konfigurace zařízení se nezachovává. Obvod LDC1101 není schopen provádět měření v jakémkoliv úsporném režimu.

2.4.2.1 Active Mode V tomto módu jsou všechny systémy obvodu spuštěny a napájeny. Probíhá měření a převod hodnot.

2.4.2.2 Sleep Mode Do Sleep módu je obvod uveden nastavením START_CONFIG.FUNC_MODE bitů na 01(registr 0x0B bity [1],[0]). V tomto módu je zachován obsah registrů. K ukončení tohoto módu a zahájení měření je potřeba nastavit dvojici bitů START_CONFIG.FUNC_MODE

na 00. Ve sleep módu je aktivní rozhraní SPI, což umožňuje čtení a zápis do registrů v tomto režimu. Při zapnutí a při opouštění Shutdown módu je LDC1101 ve Sleep módu.

Jakákoliv konfigurace zařízení musí být prováděna ve Sleep módu. Pokud je tedy potřeba za běhu programu změnit nastavení je potřeba zařízení nejprve uvést do tohoto módu a až poté provádět změny v příslušných registrech. Po provedení změn lze zařízení vrátit do aktivního módu a pokračovat ve funkci.

2.4.2.3 Shutdown Mode Shutdown mód je nejúspornější režim nabízený obvodem LDC1101. Tento režim navrátí všechny registry do jejich původní hodnoty a tudíž je potřeba všechna nastavení znovu provést při přechodu z tohoto režimu. Pro uvedení zařízení do Shutdown módu je potřeba:

1. Nastavit ALT_CONFIG.SHUTDOWN_EN(registr 0x05-bit[1]) bit na 1.
2. Přestat přivádět hodinový signál na pin CLKIN a uvést tento pin do logické 0.
3. Nastavit START_CONFIG.FUNC_MODE bity na 10(registr 0X0B bity[1],[0]). Tento registr lze nastavovat i v active módu a po splnění těchto kroků obvod přejde do Shutdown režimu.

Pro ukončení shutdown módu stačí přivést clock signál na CLKIN. LDC1101 poté přejde do sleep módu s původními hodnotami registrů. V tomto režimu neprobíhá žádné měření. Pokud kvůli nastavení registrů na původní hodnotu vznikne chyba bude tato chyba nahlášena když zařízení ukončí Shutdown mód.

2.5 Vysílač AX8052F143 [8]

Jako modul pro komunikaci se sítí Sigfox byl zvolen obvod AX-SFEU od výrobce On Semiconductor. Tento obvod byl zvolen, protože je dodáván jako součást IoT Development kitu společnosti On Semiconductor a splňuje všechny požadavky na velikost a spotřebu. AX-SFEU je jednočipový obvod pro práci na síti Sigfox funkční pro up-link i down-link při zachování velmi nízké spotřeby energie. Tento čip je dodáván se vším potřebným firmwarem k odesílání a přijímání dat ze sítě Sigfox v Evropě. Obvod AX-SFEU je dodáván ve dvou variantách.

První variantou je obvod s podporou uživatelské aplikace(API) AX-SFEU-API. Výhodou této varianty je eliminace potřeby procesoru. Nevýhodami jsou omezení v rozsahu použitého programu, omezené nástroje pro vývoj softwaru a jejich omezená podpora.

Druhou variantou je obvod komunikující s řídícím procesorem pomocí sběrnice UART. V tomto případě procesor řídí obvod pomocí AT příkazů. K této variantě existuje rozsáhlá dokumentace, jsou k dispozici různé aplikační knihovny a velká možnost výběru procesorů a programátorských rozhraní. V tomto projektu byla tudíž využitá druhá varianta AX-SFEU.

Základní vlastnosti obvodu AX-Sigfox[8]:

- obvod certifikovaný pro síť SigFox
- napájení 2.1 V - 3.6 V
- provozní teplota -40 °C - 85 °C
- meření teploty a napájecího napětí
- spotřeba v deep sleep módu: 100 nA
- spotřeba v sleep módu: 1.3 uA
- spotřeba v standby módu: 0.5 mA
- spotřeba při vysílání 10 mA

2.5.1 Standby mode

Po zapnutí a dokončení přenosu po síti SigFox tento obvod zůstává ve Standby režimu. Obvod naslouchá komunikaci na UART sběrnici a očekává příkazy. Spotřeba v tomto módu činí 0.5 mA. Pro šetření energie je možno obvod uvést do Sleep nebo Deep Sleep módu.

2.5.2 Sleep mode

Příkaz AT$P=1 uvede zařízení do Sleep módu. V tomto módu stále běží časovač pro out of band(Zprávy, kterou každé Sigfox zařízení vysílá periodicky, tato zpráva obsahuje informace jako teplotu zařízení a napětí baterie. Tyto zprávy se nepočítají do denního limitu odeslaných zpráv.) zprávy. Pro probuzení z tohoto módu stačí signalizovat Rx pin posláním přestávky.(Přestávka je definována jako alespoň 10 po sobě jdoucích bitů s hodnotou logické 0 a taková sekvence bitů je proti pravidlům RS232 komunikace). Při odesílání out of band zprávy se obvod automaticky převede do přenosového režimu, odešle zprávu a poté se uvede zpátky do režimu Sleep. Spotřeba zařízení v tomto módu činí 1.3 uA.

2.5.3 Deep sleep mode

V tomto módu je zařízení úplně vypnuté a vyžaduje jen zanedbatelné množství energie. Aktivace Deep Sleep módu probíhá zasláním příkazu AT$P=2. K probuzení z tohoto módu je potřeba pin GPIO9 uvést do logické nuly. Při použití tohoto módu se vypnou všechny časovače a paměti. Pokud chce uživatel použít nastavení uložená v registrech může tak učinit příkazem AT$WR, který uloží data z registrů do flash paměti. Spotřeba zařízení v tomto módu činí 100nA.

3 Síť IoT Sigfox

IoT je technologie využívaná převážně u zařízení vyžadujících dlouhý běh s nízkou spotřebou, která nepotřebují odesílat velké množství dat. Pro správný běh aplikace v řádu měsíců až let je potřeba dosáhnout celkové spotřeby v jednotkách až setinách mA.

Nízké spotřeby je docíleno výběrem úsporných procesorů a ostatních komponent, které nabízejí nízkou spotřebu v aktivním režimu a současně jsou schopné běžet ve velmi úsporných stand-by režimech. V těchto standby režimech bývá aktivní jen nutné minimum obvodů. K přechodu obvodů do aktivního režimu probíhá pouze pokud to aplikace potřebuje na minimální časové úseky, jinak obvody zůstávají v režimech s nízkou spotřebou.

V dnešní době je převážná většina procesorů používaných v IoT aplikacích postavena na jádru ARM[10]. Výhodou je jednoduchý instrukční set pro používané procesory a tím pádem velmi dobrá přenositelnost programů a knihoven mezi různými typy procesorů. Tímto se značně zjednodušuje a urychluje návrh a vývoj aplikací. Nevýhodami této jednotné architektury mohou být poněkud zvýšená spotřeba energie a zvýšená komplikovanost konečného systému.

3.1 Technologie IoT Sigfox

Jako IoT síť byla použita síť Sigfox. Sigfox má vybudované zázemí jak na straně procesoru, kde má pro komunikaci po síti již vytvořené různé aplikační knihovny, tak na straně sítě, kde umožňuje vytváření callbacků a navázání zasílaných zpráv do různých nadřazených systémů. Tato síť má neustále se rozšiřující pokrytí ve většině evropských zemí, má velmi dobrou odolnost vůči rušení a jiným vnějším vlivům a vlastní zabezpečení, které je na velmi dobré úrovni.

Na území české republiky je sigfox provozován firmou Simplecell Networks a.s.

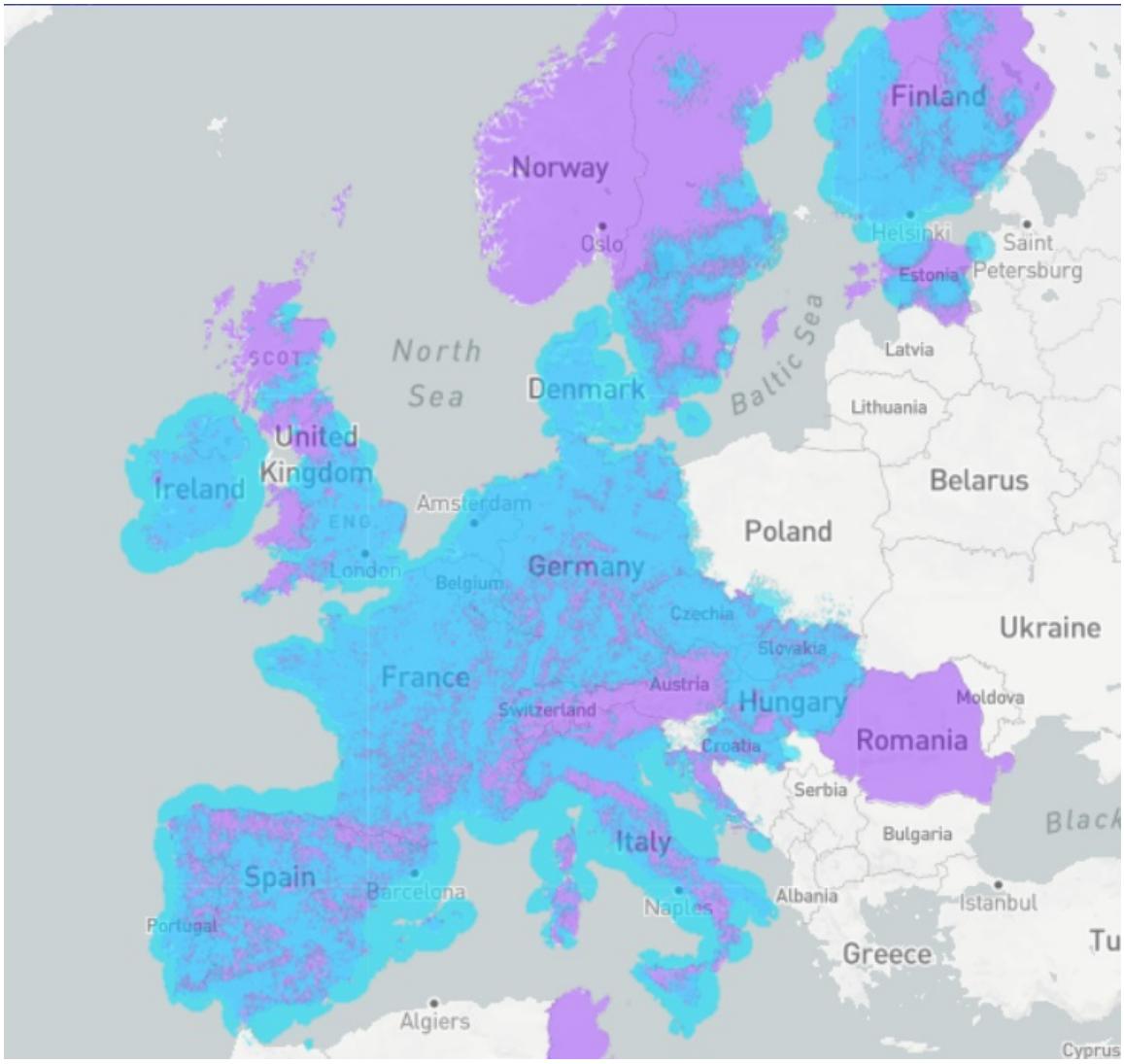
Sigfox využívá rádiovou technologii Ultra Narrow Band a operuje v nelicencovaných pásmech. Jednotlivé zprávy zpracovávané sítí Sigfox jsou malé (12B uživatelských dat na zprávu + 1B hlavička.).

Síť Sigfox má omezené množství odeslaných zpráv. Maximální povolený počet zpráv od zařízení na cloud je 6 za hodinu, tedy 144 zpráv denně. Tato zpráva dokáže pojmout maximálně 12B uživatelských dat. Při odesílání zpráv ze sítě na zařízení je povoleno odesílat 4 zprávy denně. Tato zpráva může obsahovat maximálně 8B dat.

3.2 Backend Sigfox

Sigfox má již vytvořené webové rozhraní pro práci se zprávami přijatými ze zařízení. Rozhraní je dostupné na adrese `backend.sigfox.com`.

Toto rozhraní umožňuje vytvářet a spravovat skupiny uživatelů a těm přidělovat práva k daným zařízením v síti Sigfox. Lze zde prohlížet jednotlivá zařízení, jejich odeslané zprávy, vytvářet callbacky a kontrolovat případné chyby při odesílání zprávy ze zařízení na Sigfox. Je umožněno také sledovat polohu zařízení, tato služba je však pouze orientačního charakteru, odchylka se



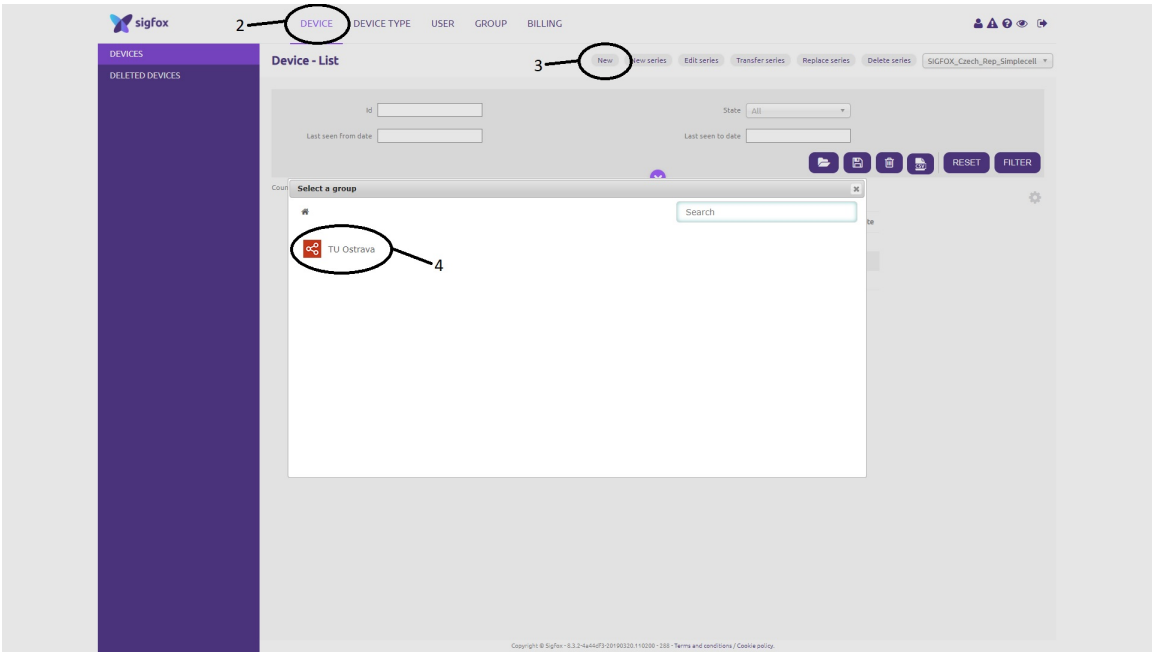
Obrázek 4: Pokrytí sítí Sigfox v Evropě(Duben 2019) `https://www.sigfox.com/en/coverage`

● - Aktuální pokrytí ● - Síť ve výstavbě

totiž pohybuje v řádech desítek km.

Pro registraci nového zařízení (Obr. 5) je potřeba provést následující kroky:

1. Přihlásit se pod svým účtem na adrese `backend.sigfox.com`.
2. V horní liště zvolit menu *Device*.
3. V pravé horní části obrazovky zvolit možnost *New*.
4. Vybrat požadovanou skupinu.
5. Vyplnit informace o zařízení (ID/PAC, Name, Device Type, Product certificate, Základní stav je *Activable*, pokud je tato možnost zakázána všechny zprávy budou zahozeny. Tuto možnost lze nastavit v editaci zařízení.)
6. Potvrzení tlačítkem *Ok*.



Obrázek 5: registrace Sigfox zařízení

Pro vytvoření vlastního callbacku je potřeba provést následující kroky:

1. Přihlásit se pod svým účtem na adrese `backend.sigfox.com`.
2. V horní liště zvolit menu *Device Type*.
3. Spravovat Device type a nastavit *Downlink mode* na *callback*.
4. V *Callbacks* menu vytvořit *custom callback*.
5. Nastavit:
   - Type: DATA/BIDIR
   - Channel: URL
   - Url pattern: doména + umístění (`http://host/path`)
   - Zvolit možnost POST.
   - Content type: application/json
   - Body: Data, která budou odeslána na zařízení, nastavitelná dle potřeby.

4 Implementace

4.1 Návrh řešení

Požadavkem je, aby program běžel po dobu několika měsíců. Celý program jsem realizoval jako nekonečnou smyčku. Kvůli úspoře energie bude procesor po většinu času v režimu pro šetření spotřeby, ve kterém se program nevykonává vůbec, v paměti je však uchován úsek, kde byl program před přechodem do úsporného režimu. První věc, kterou program provede po startu je právě přechod do úsporného režimu. Probuzení z tohoto režimu může nastat dvěmi různými možnostmi. První je přerušení časovačem pro kontrolu správného nošení. Přerušení pomocí časovače bude nastávat periodicky co 15 minut. Druhou možností je buzení pomocí vnějšího přerušení pro provedení hlavního měření a následné odeslání dat do sítě Sigfox. Toto přerušení je potřeba aktivovat ručně, prakticky je provedeno přiložením magnetu ke spínacímu čidlu.

4.1.1 Přerušení časovačem

Při přerušení časovačem probíhá pouze kontrola správnosti nošení. K této kontrole slouží senzor LDC1101, ten provede měření vzdálenosti, jehož výsledek porovná s referenční hodnotou. Není zde potřeba ukládat celý výsledek měření. Při každém tomto měření se inkrementuje čítač celkových kontrolních měření. Pokud helma nedetekuje hlavu dojde dále k inkrementaci čítače nenošených měření.

4.1.2 Ruční přerušení

Při ručním přerušení program provede měření vzdálenosti pomocí senzoru LDC1101 a uloží celý výsledek tohoto měření. Po dokončení tohoto měření začne program odesílat naměřená data společně s daty o nošení do sítě Sigfox. Po odeslání program čeká jednu minutu na ack signál, který potvrzuje úspěšné přijetí dat na Sigfox. Pokud je tento signál obdržen program pokračuje přechodem do úsporného režimu. V případě neobdržení ack zprávy je pro danou zprávu nastaven příznak a při příštím odesílání dat dojde k jejímu opětovnému odeslání. K signalizaci komunikace se sítí Sigfox je v obvodu zabudovaná LED dioda, která svítí pokud probíhá komunikace.

4.1.3 Formát dat odesílaných do sítě Sigfox

Formát zprávy zůstává fixní po celou dobu provozu zařízení.
{part}{time}{data}{uncovered}

- Část **{part}(1B)** jsou data odeslaná formou jednoho 8b čísla (uint_8). Tato data určují o kolikáté měření vzdálenosti od nasazení zařízení se jedná. Pořadí se počítá od 1.

- Část **{time}(2B)** jsou data odeslaná formou 16b čísla (uint_16). Tato data určují počet uběhlých kontrolních 15 minutových intervalů od nasazení zařízení do provozu.
- Část **{data}(3B)** jsou data odeslaná formou 16b čísla (uint_16). Pro tuto část je rezervovaný 1B navíc pro případné budoucí rozšíření na měřící mód LHR, který je přesnější. Jsou to data naměřená senzorem LDC1101 udávající vzdálenost lebky od helmy. Jedná se o data přímo vracená senzorem, k jejich zpracování dojde až na serveru.
- Část **{uncovered}(2B)** jsou data odeslaná formou 16b čísla (uint_16). Data obsahují počet kontrolních intervalů od posledního odeslání zprávy, ve kterých byla helma detekovaná jako nenošená.

```
struct Data {
   uint8_t MeasureCounter;
   uint16_t ControlCounter;
   uint16_t rawData;
   uint16_t NoWear;
 };
```

Výpis 1: Struktura dat odesílaných do sítě Sigfox.

4.2 Mbed [11]

Prvotní vývoj programu jsem realizoval na platformě Mbed. Tuto platformu jsem zvolil, protože byla doporučena výrobcem prvního procesoru NCS36510, na kterém probíhal vývoj. Mbed je open-source operační systém[12] přímo tvořený pro implementaci IoT aplikací na mikroprocesorech. Mbed nabízí především spoustu aplikačních knihoven pro vývoj v jazyce C/C++. Jakékoliv knihovny v rámci mbedu lze použít ke komerčním účelům bez potřeby platit za jakékoliv licence.

Za pomocí tohoto projektu lze vyvíjet aplikace na Arm Cortex-M procesorech a součástí projektu jsou knihovny zajišťující práci s různými periferiemi. Tento projekt zajišťuje jednoduchou přenositelnost již existujících programů mezi různými procesory a významně zjednodušuje a zrychluje vývoj programů nových.

Mbed nabízí vlastní online IDE, ale také možnost používat mbed OS v rámci jiných IDE, pokud si tuto možnost zvolí programátor. Programování samotného procesoru je možné pomocí USB a umožňuje jednoduché a levné prototypování.

4.3 Arduino [6]

Pro implementaci finálního programu jsem využil platformu Arduino. Hlavním důvodem této volby byla již ověřená práce s použitým procesorem SAMD21 na této platformě. Arduino je open-source platforma zakládající si na jednoduchosti použitého hardwaru a softwaru.

Platforma Arduino byla zpočátku vytvořena jako nástroj pro rychlé a jednoduché prototypování sloužící především studentům bez hlubších znalostí elektroniky a programování. S postupně se rozšiřující komunitou uživatelů se platforma Arduino také musela postupně adaptovat potřebám stále náročnějších uživatelů. Z jednoduchých 8-bitových aplikací se tak platforma rozrostla pro možnosti práce s IoT, 3D tiskem a jinými embedded zařízeními. Všechny desky Arduino jsou open-source, čímž umožňují uživatelům sestavovat je nezávisle a adaptovat je ke svým potřebám. Software je také kompletně open-source a nabízí spoustu aplikačních knihoven vyvíjených velkou komunitou nadšenců z celého světa.

Programovací jazyk Arduino je rozšíření jazyka AVR C, které podporuje i jazyk C++ a jeho knihovny. Jazyk obsahuje základní funkce pro práce s MCU jako jsou například čteni a zápis na sériové komunikaci nebo nastavování digitálních pinů. Program je strukturován do dvou hlavních částí. První částí je funkce setup sloužící k prvotnímu nastavení programu, tato funkce se vykoná jednou na začátku programu. Druhou částí je funkce loop, tato funkce se začne vykonávat po proběhnutí funkce setup a funguje jako nekonečně běžící smyčka.

Hlavními výhodami platformy Arduino jsou:

- Nízká cena - Platformy Arduino jsou v porovnání s ostatními MCU platformami relativně levné, hotové vývojové desky lze u nás zakoupit pod 500 Kč.
- Přenositelnost - Vývojové prostředí Arduino IDE je k dispozici na operačních systémech Windows, Linux i Macintosh OSX
- Jednoduché programovací prostředí - Arduino IDE je velmi jednoduché pro použití i pro naprosté začátečníky. Zároveň je dostatečně flexibilní pro uspokojení náročnějších a zkušenějších programátorů.
- Open source a rozšiřitelný software - Všechen software pro Arduino je publikován jako Open source. Jazyk lze rozšířit o libovolné C++ knihovny. Jazyk lze také použít současně s programovacím jazykem AVR C, na jehož základech je platforma Arduino vyvíjena. Je možné také použití AVR C jazyka přímo v Arduino projektu.
- Open source a rozšiřitelný hardware - Návrhy Arduino desek jsou publikovány, takže zkušení návrháři obvodů mohou vytvářet, rozšiřovat a vylepšovat jejich vlastní verze modulů. I nezkušení návrháři si mohou vytvořit vlastní obvod na nepájivém poli a tím porozumět co jak funguje.

4.4 Komunikace po síti Sigfox

Na použité platformě Arduino je implementace UART komunikace se zařízením AX-SFEU velmi jednoduchá. Použitý vývojový kit SAMD21 mini má určené piny pro UART komunikaci(RX a TX), po připojení zařízení na tyto piny je potřeba ve funkci setup inicializovat komunikaci. Inicializace probíhá jednoduše pomocí příkazu Serial1.begin(Baud rate), kde Baud rate je přenosová rychlost, tedy počet prvků přenesených za sekundu. Modul AX-SFEU má modulační rychlost 9600 baud. Pro sériovou komunikaci Arduino nabízí základní funkce[13] jako například read, write nebo print.

```
// the setup function runs once when you press reset or power the board
void setup() {
  Serial1.begin(9600); //initialize serial communication
}

// the loop function runs over and over again forever
void loop() {
  Serial1.write(byte(0x00)); //wake AX-SFEU up from sleep mode
  Serial1.write("AT$SF=48656c6c6f20776f726c6421"); // AT Command for
      transmission
  Delay(5000);
 }
```

Výpis 2: Odesílání zprávy "Hello world!"na Sigfox v pětisekundových intervalech pomocí modulu AX-SFEU.

4.4.1 AT Příkazy

Celé ovládání zařízení AX-SFEU probíhá pomocí AT příkazů. Každý takový příkaz začíná prefixem 'AT', pokračuje příkazem s parametry pokud je příkaz požaduje a končí jakýmkoliv prázdným znakem. V případě nespravné syntaxe příkazu je celý vstup do následujícího prázdného znaku ignorován. Po správném vykonání příkazu modul na rozhraní UART odesílá zprávu ve tvaru 'OK'. Pokud má příkaz návratovou hodnotu je odeslána pouze tato hodnota.

Například:

Procesor odešle: **AT$SF=aabb1234**
Modul odešle Sigfox rámec obsahující 0xaa : 0xbb : 0x12 : 0x34 bez čekání na odpověď.
Modul odešle odpověď procesoru: OK

Procesor odešle: **AT$SF=0011223344,1**
Modul odešle Sigfox rámec obsahující 0x00 : 0x11 : 0x22 : 0x33 : 0x44 a vyčkává na odpověď, kterou po obdržení odešle pomocí UART.
Modul odešle odpověď procesoru: OK
Modul odešle odpověď procesoru: RX = AA BB CC DD

Procesor odešle: **AT$P=1**
Procesor je převeden do režimu sleep.
Modul odešle odpověď procesoru: OK

Tabulka 2: Přehled použitých AT příkazů

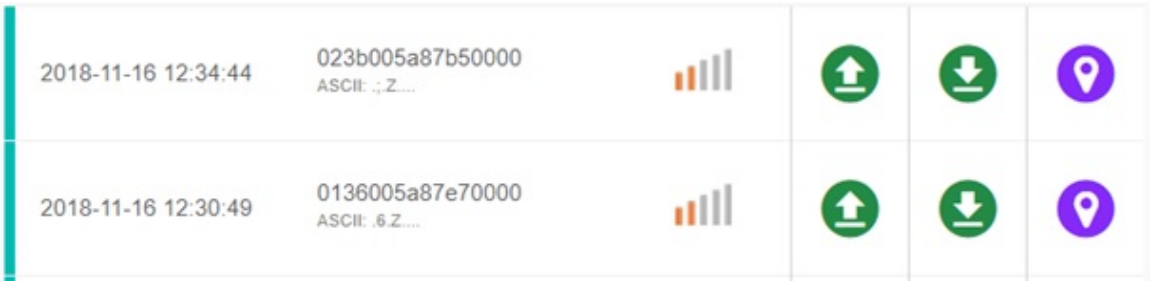
Příkaz	Název	Popis
AT$SF=frame	Send Frame	Odešle data, 1B - 12B. Nepovinný bit indikuje, zda má AX-SFEU očekávat downlink zprávu.
AT$uint=uint	Set Register	Nastavení konfiguračního registru.
AT$V?	Get Voltages	Vrátí aktuální napětí a napětí naměřené při posledním vysílání v mV
AT$P=uint	Set Power Mode	Převedení zařízení do úsporného režimu. 0: Software reset 1: sleep 2:deep sleep
AT$WR	Save Config	Zapíše všechny nastavení z registrů do flash paměti. Při použití AT$P=0 dojde k automatickému načtení nastavení z flash paměti.

4.4.2 Reprezentace dat na straně sítě Sigfox

Dáta jsou do sítě Sigfox doručovány v rámcích obsahujících pořadové číslo a data jako posloupnost bitů. Ve výchozím stavu jsou tato data reprezentována jako hexadecimální číslo, ale lze nastavit i jinou reprezentaci jako například Ascii symboly nebo vlastní reprezentaci příchozích dat.

Pro zobrazovaní zpráv je na adrese `backend.sigfox.com` k dispozici uživatelské rozhraní. Po registraci zařízení do sítě Sigfox se zde pro zařízení automaticky ukládají všechny odeslané zprávy. U každé zprávy (obr. 6) lze kromě samotného obsahu zobrazit také kvalitu signálu při odesílání dané zprávy, status callbacků a lokaci zařízení v době odesílání zprávy. Služba lokace zařízení je však pouze orientačního charakteru, jelikož odchylka se zde pohybuje v desítkách km.

Rozhraní Sigfox umožňuje exportovat zprávy včetně detailních informací jako například pořadové číslo zprávy nebo čas doručení zprávy. Exportovat data lze přímo do csv souboru nebo lze nastavit callback na odeslání dat na jiné platformy, mezi tyto platformy patří například cloud služba amazon web services, která byla také v rámci projektu využita.



Obrázek 6: Zobrazení dat na platformě `backend.sigfox.com`

Zde jsou znázorněny interpretace konkrétních dat ve formátu popsaném v kapitole 4.1.3.
Popis:
{part} = 1 (0x01); **{time}** = 13824 (0x3600) **{data}** = 0035627 (0x5a87e7) **{uncovered}** = 0 (0x00)
Tuto zprávu lze interpretovat následujícím způsobem:

- Pořadové číslo měření je 1, tedy proběhlo první měření, které bylo odesláno do sítě Sigfox.
- Od začátku nošení přilby uběhlo 13 824 časových bloků (1 blok = 15 min.).
- Hodnota naměřena na senzoru (RAW DATA) je 35627.
- Počet časových bloků, kdy přilba nebyla nasazena od posledního měření odeslaného na Sigfox je 0 (1 blok = 15 min.)

Time	Type	Severity	Source id	Description	Status
2018-08-11 19:38:40	Break in message sequence	WARN	22AB7F	Break in message sequence from Device #22AB7F [396 inferior 398]	
2018-08-10 10:38:58	Break in message sequence	WARN	22AB7F	Break in message sequence from Device #22AB7F [392 inferior 396]	
2018-08-07 15:34:05	Break in message sequence	WARN	22AB7F	Break in message sequence from Device #22AB7F [384 inferior 390]	
2018-07-10 14:40:25	Break in message sequence	WARN	22AB7F	Break in message sequence from Device #22AB7F [382 inferior 384]	
2018-07-10 14:39:50	Break in message sequence	WARN	22AB7F	Break in message sequence from Device #22AB7F [377 inferior 381]	

Obrázek 7: Tabulka events na `backend.sigfox.com` zobrazující přerušení v sekvencích zpráv.

4.4.3 Zajištění kontinuity dat

Síť Sigfox nijak nezaručuje garantované doručení zpráv. Proto se především v oblastech s horším signálem může stát, že zpráva po odeslání nedorazí do sítě Sigfox. Každé zařízení pro komunikaci má proto pro zprávy zavedené pořadové číslo, které je odesíláno v její hlavičce. Při správném doručení zprávy na server tak lze detekovat případné výpadky dřívějších zpráv. Nejjednodušší způsob monitorování těchto výpadků je přímo v záložce events pro zařízení (obr. 7).

Pro zajištění správného doručení zpráv umožňuje Sígfox odesílání downlink zpráv do zařízení. Nejjednodušší formou downlink zprávy je odeslání ACK zprávy, která potvrzuje správné doručení rámce. V zařízení se vždy zpráva ukládá v datové struktuře a má příznakový bit, který značí, zda byla daná zpráva správně odeslána. Pokud zařízení neobdrží ACK zprávu po pokusu o odeslání dat nastaví příznak pro danou zprávu a nepokouší se dále odesílat data. Pokud proběhně odeslání dat úspěšně zařízení se postupně pokusí odeslat všechny zprávy označené jako neodeslané.

4.5 Vyčítání hodnot z čidla indukčnosti

4.5.1 SPI Komunikace

Veškerá komunikace mezi mikroprocesorem a čidlem LDC1101 probíhá po rozhraní SPI. SPI je synchronní sériový protokol pro přenos dat používán prioritně u mikropočítačů pro komunikaci s jedním nebo více periférními zařízeními. V zapojení SPI je vždy jedno zařízení Master, které ovládá všechna ostatní zařízení. Typicky jsou s každým zařízením propojeny 4 datové kanály:

- **MISO** (Master In Slave Out) - Kanál, kterým Slave zařízení odesílá data zařízení Master.
- **MOSI** (Master Out Slave In) - Kanál, kterým Master zařízení odesílá data zařízením Slave.
- **SCK** (Serial Clock) - Hodinový signál, který se stará o synchronizaci datového přenosu.
- **SS** (Slave Select) - Kanál, kterým Master umožňuje a znemožňuje komunikaci specifickým Slave zařízením.

Arduino nabízí knihovnu pro práci s SPI[14]. Pro práci s touto knihovnou je nejprve potřeba shromáždit několik informací o zařízení. První potřebnou informací je maximální frekvence SPI na které zařízení může fungovat. Druhou potřebnou informací je způsob řazení dat, tedy zda je prvním bitem MSB(Most Significant Bit) nebo LSB(Least Significant Bit). Třetí a poslední potřebnou informace sděluje, jesli je hodinový signál pro data nečinný v logické 1 nebo logické 0. Tyto parametry se nastavují v SPISettings.

Obvod LDC1101 používá rozhraní SPI ke konfiguraci vnitřních registrů. Konfiguraci zařízení LDC1101 je nutné provádět v režimu Sleep. Při změně nastavení je tedy potřeba zařízení nejprve převést do režimu Sleep, v tomto režimu změnit požadovaný registr a následně vrátit LDC1101 do aktivního režimu. Na SS musí být přivedena logická 0 před přístupem k první adrese. Příkaz k přepsání registru změní obsah adresovaného registru jen pokud je při jeho vykonávání souběžně odesláno alespoň 16 SCK pulsů (obr. 8). Obvod LDC1101 také podporuje prodloužené SPI transakce, u kterých je SS držen v logické 0 a následující adresy registrů mohou provádět čtení nebo zápis. Po první operaci na registru je s každými následujícími 8 SCK pulsy adresován následující registr, na kterém může probíhat čtení nebo zápis dle nastavení v prvotním příkazu. Touto metodou lze programovat 2 nebo více registrů. Adresa registru programovaného prodlouženou SPI transakcí nesmí přesáhnout hodnotu 0x3F.

```
#include<SPI.h>

int LDCSlaveSelect = 2;
SPISettings LDC1101(1000000,MSBFIRST,SPI_MODE3);

void setup() {
```

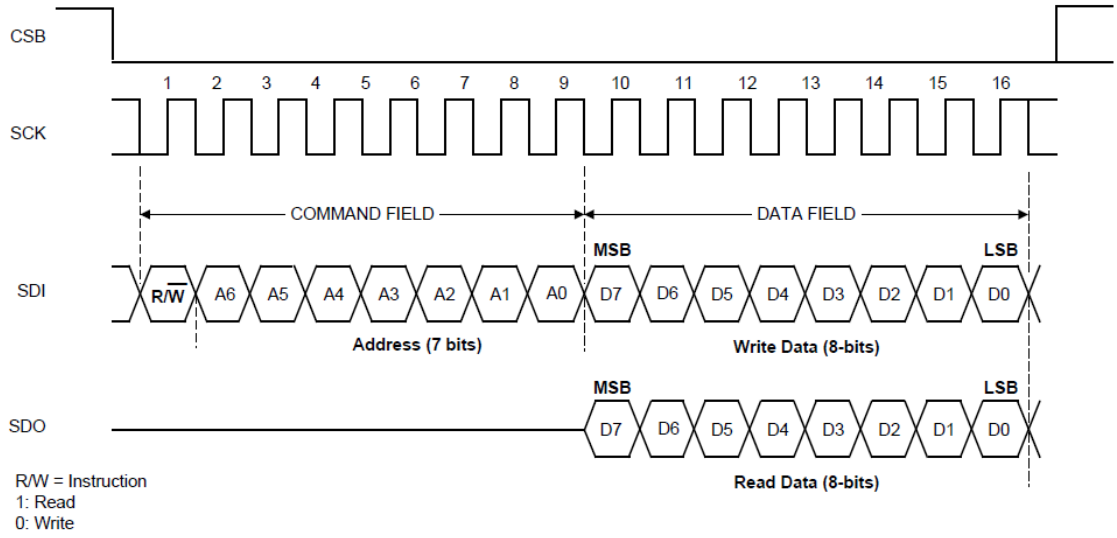
```
  pinMode(LDCSlaveSelect, OUTPUT);
  SPI.begin();
  Serial.begin(9600);
  SPI.beginTransaction(LDC1101);
  writeSPI(0x05, 0x03); // RP mode + Shutdown enabled
  writeSPI(0x0C, 0x01); // Enable RP, Disable LHR
  SPI.endTransaction();
}

void loop() {
  uint16_t RPData;
  RPData = readSPI(0x21) | (readSPI(0x22)<<8);
  Serial.println(RPData);
  Delay(1000);
}

void writeSPI(byte Register, byte Value)
{
  SPI.beginTransaction(LDC1101);
  digitalWrite(LDCSlaveSelect, LOW);
  SPI.transfer(Register);
  SPI.transfer(Value);
  digitalWrite(LDCSlaveSelect, HIGH);
  SPI.endTransaction();
}

unsigned int readSPI (byte Register)
{
  unsigned int result = 0;
  SPI.beginTransaction(LDC1101);
  digitalWrite(LDCSlaveSelect, LOW);
  SPI.transfer(Register | 0x80);
  result = SPI.transfer(0xFF);
  digitalWrite(LDCSlaveSelect, HIGH);
  SPI.endTransaction();
  return result;
}
```

Výpis 3: Měření vzdálenosti každou vteřinu pomocí LDC1101.



Obrázek 8: Formát SPI Transakce zařízení LDC1101[9]

Tabulka 3: Přehled registrů použitých při práci s obvodem LDC1101

Adresa	Název	Výchozí hodnota	Popis
0x01	RP_SET	0x07	Konfigurace dynamické vzdálenosti Rp měření.
0x05	ALT_CONFIG	0x00	Konfigurace nastavení zařízení
0x0C	D_CONF	0x00	Vyžadování kontrolní amplitudy čidla
0x21	RP_DATA_LSB	0x00	Výsledná data pro Rp měření, bity 7:0
0x22	RP_DATA_MSB	0x00	Výsledná data pro Rp měření, bity 15:8

Tabulka 4: Naměřené hodnoty při vzdálenosti vodivého terče od senzoru.

Vzdálenost	R_p	R_{pDATA}
0mm	1.25kΩ	128
1mm	1.87kΩ	14200
2mm	2.79kΩ	29015
3mm	4.06kΩ	40600
4mm	7.92kΩ	53515
5mm	10kΩ	56198

4.5.2 Převedení naměřených hodnot na reálnou vzdálenost

Pro měření vzdálenosti jsem použil režim Rp+L. V tomto režimu čidlo LDC1101 současně měří impedanci a rezonantní frekvenci připojeného senzoru. Pro výpočet vzdálenosti je potřeba hodnota impedance Rp. Obvod LDC1101 je schopen vrátit 16b digitální hodnotu úměrnou impendanci Rp. Výpočet hodnoty impedance na základě dat je v rovnici (2).

$$R_P = \frac{R_{pMAX} * R_{pMIN}}{R_{pMAX} * (1 - \frac{R_{pDATA}}{2^{16}-1})} + R_{pMIN} * \frac{R_{pDATA}}{2^{16}-1} \tag{2}$$

- R_p - Impedance Rp.
- R_{pMAX} - Maximální možná hodnota impedance Rp pro daný systém (96kΩ).
- R_{pMIN} - Minimální možná hodnota impedance Rp pro daný systém (1.25kΩ).
- R_{pDATA} - Digitální hodnota naměřená senzorem.

Dále je potřeba převést impedanci Rp na vzdálenost. Protože impedanci Rp kromě vzdálenosti ovlivňují i faktory jako velikost a složení vodivého materiálu musel jsem závislost Rp na vzdálenosti od materiálu odvodit experimentálním porovnáváním naměřených hodnot a skutečné vzdálenosti. Závislost vzdálenosti na naměřencých datech se jeví téměř lineárně. Při přiložení senzoru k vodivé ploše je naměřená hodnota R_pDATA 128, poté se postupně s rostoucí vzdáleností zvětšuje, u vzdálenosti 5mm už se tato hodnota pohybuje přes 56000, což je hodnota naměřená i po úplném odebrání vodivých materiálů z blízkosti senzoru. Konečný přibližný vztah pro zjištění vzdálenosti z naměřených dat je:

$$l = \frac{R_{pDATA} - 128}{14000} \tag{3}$$

4.6 Optimalizace spotřeby

Spotřeba byla optimalizovaná použitím úsporných režimů u všech využitých komponent a také prováděním pouze opravdu potřebných úloh na straně procesoru. Všechny úlohy, které je možné provádět až později jako například převod Rp dat na vzdálenost jsou prováděny až z dat obdržených na síti Sigfox. Díky minimalizaci úloh prováděných na procesoru je tak možné udržet zařízení v úsporných režimech po delší dobu.

4.6.1 Obsluha úsporných režimů

K práci s úsporným režimem na procesoru SAMD21 jsem využil knihovnu RTCZero[15] pro Arduino. Tato knihovna umožňuje jednoduché používání RTC generátoru k časování přechodu procesoru z režimu Standby do aktivního režimu. Při provádění kontroly nošení přilby způsobeném přerušením časovačem poté aplikace pouze krátce převede čidlo LDC1101 do měřícího režimu, provede jedno měření a uloží informace o nošení, poté dojde k uspání čidla a následně k opětovnému uspání procesoru. Při provádění hlavního měření způsobeném externím přerušením dojde k převodu čidla LDC1101 do měřícího režimu a uložení celého výsledku měření. Poté je čidlo uvedeno zpátky do úsporného režimu a dochází k aktivaci obvodu AX-SFEU, který odesílá data, u každé zprávy čeká minutu na potvrzení doručení. Poté je obvod AX-SFEU převeden do úsporného režimu a nakonec je do úsporného režimu převeden i procesor.

```
#include <RTCZero.h>

RTCZero rtc;

const byte interruptPin = 7;
const byte timeInterval = 15;

void setup() {

  pinMode(interruptPin, INPUT_PULLUP);

  rtc.begin();
  rtc.setTime(0, 0, 0);
  rtc.setDate(0, 0, 0);
  rtc.setAlarmMinutes(15);
  rtc.enableAlarm(rtc.MMSS);

  rtc.attachInterrupt(controlMeasurement);
```

```
  rtc.attachInterrupt(digitalPinToInterrupt(interruptPin), detailedMeasurement,
      HIGH);

  rtc.standbyMode();
}

void loop() {
 rtc.setAlarmMinutes(rtc.getMinutes() + timeInterval);
 rtc.standbyMode();
}
```

Výpis 4: Obsluha hodinového a externího přerušení procesoru SAMD21.

4.6.2 Výpočet spotřeby energie

Kromě dříve popsaných obvodů je dále požadována jedna signalizační LED dioda sloužící k signalizaci komunikace se sítí Sigfox. Spotřeba této diody při provozu činí 10mA.

Celkový očekávaný čas zařízení v režimu s nízkou spotřebou jsou 4 měsíce, tedy $4 * 30 * 24 = 2880$ hodin. Za 2880 hodin je očekávaná spotřeba $22.7 * 10^{-6}A * 2880h \doteq \underline{0.065Ah}$.

U kontrolního měření předpokládám dobu trvání na 0.25s a provádění tohoto měření každých 15 minut. Za 4 měsíce provozu to tedy znamená $0.25s * 4 * 2880 = 2880s = 0.8h$ v tomto režimu. Spotřeba energie při kontrolním měření za 4 měsíce činí $38 * 10^{-3}A * 0.8 \doteq \underline{0.03Ah}$.

U detailního měření předpokládám dobu trvání na 0.25s a provádění tohoto měření jednou denně. Za 4 měsíce provozu bude zařízení v detailním měření po dobu $0.25s * 120 = 30s \doteq 0.01h$. Celková spotřeba energie při detailním měření za 4 měsíce je $38 * 10^{-3}A * 0.01 \doteq \underline{0.0004Ah}$.

U odesílání zprávy předpokládám dobu trvání zhruba 45s. Odesílání zprávy bude probíhat jednou denně. Doba trvání je značně závislá na stavu sítě a síle signálu v době odesílání zprávy. Uvažována je nejhorší varianta. Celkový čas odesílání při 4 měsičním provozu činí $120 * 45s = 5400s = 1.5h$. Celková spotřeba obvodů za 4 měsíce $40 * 10^{-3}A * 1.5 = 0.06Ah$. Spotřeba signalizační LED diody činí $10 * 10^{-3}A * 1.5h = 0.015Ah$. Celková spotřeba energie při odesílání zpráv během 4 měsíců očekávaného používání aplikace tak činí $0.06Ah + 0.015Ah = \underline{0.075Ah}$.

Kalkulace všech spotřeb byla záměrně provedena na pro dobu delší než je zamýšlená doba používání a je v ní počítáno s nejhoršími variantami z pohledu spotřeby jednotlivých součástek. $0.065Ah + 0.03Ah + 0.0004Ah + 0.075Ah = \underline{0.1704Ah}$. Odhad minimální kapacity baterie činí 170mAh.

Tabulka 5: Spotřeba jednotlivých obvodů

Obvod	Úsporný režim	Kontrolní měření	Detailní měření	Odesílání zprávy
SAMD21	20uA	30mA	30mA	30mA
LDC1101	1.4uA	1.9mA	1.9mA	1.4uA
AX-SFEU	1.3uA	1.3uA	1.3uA	10mA
Celkem	22.7uA	38mA	38mA	40mA

5 Závěr

Cílem této práce bylo vytvoření softwaru, který současně dokáže monitorovat růst lebky a kontrolovat správné nošení speciální helmy pro děti trpící plagiocefálií. Je potřeba, aby software mohl na dodaném hardwaru fungovat několik měsíců bez potřeby vyměnit baterii.

Prvním krokem bylo zprovoznění komunikace po síti Sigfox. Komunikace se sítí Sigfox probíhala pomocí certifikovaného obvodu AX-SFEU, který jsem oblushoval pomocí AT příkazů v rozhraní UART. Pro reprezentaci dat má Sigfox vytvořen velmi propracované uživatelské rozhraní, které zobrazuje všechnu zaznamenanou komunikaci s registrovaným zařízením. Také je zde možnost zasílání různých callbacků. Byl využit callback na službu Amazon Web Services využívanou zadavatelem práce.

Dále bylo potřeba zprovoznit měření vzdálenosti mezi čidlem indukčnosti LDC1101 a kovovým terčem. Obsluha čidla LDC1101 probíhala po sběrnici SPI. Zde bylo potřeba nejdřív v registrech nastavit požadovaný druh měření a následně lze při aktivním měření z registrů vyčítat naměřené hodnoty. Naměřené hodnoty jsou pouze digitální data udávající Impedanci, tyto data jsem nejdříve převedl na impedanci a následně jsem z impedance určil vzdálenost. Kvůli náchylnosti k různým vlivům jako jsou povrch či náklon terče je doporučeno provádět převod naměřených hodnot na reálnou vzdálenost po otestování dat. Proto jsem provedl testovací měření, kdy jsem postupně terč oddaloval a zaznamenával změny v naměřených hodnotách.

Data naměřená čidlem LDC1101 byla do sítě Sigfox odesílaná v "Raw"podobě, tedy jako číslo vyčtené přímo z registru senzoru. Dále bylo potřeba odesílat informace o nošení helmy, času měření a pořadí měření. Všechna data jsem ukládal do jedné struktury tak, aby bylo možno údaje o každém měření odeslat jednou zprávou.

Optimalizaci spotřeby jsem řešil především používáním nejúspornějších dostupných režimů pro všechny obvody. Do aktivního režimu jednotlivé části převádím pouze pokud je to potřeba na co nejkratší dobu. Dále je na straně procesoru prováděn minimální počet operací a všechny operace, u kterých je to možné jsou prováděny až na webovém rozhraní Sigfoxu a ostatních služeb, které přijímají data od Sigfoxu.

Při vývoji bylo potřeba postupně použít 3 různé procesory kvůli slabinám, na které se přišlo při vývoji produktu. První procesor NCS36510 dodávaný s vývojovým IoT kitem sice splňoval všechny požadavky na něj kladené, ale samotný procesor bez vývojového kitu byl prodáván v minimálním množství tisíce kusů, což je pro vývoj nepřípustné. Druhý procesor LPC824 již bylo možno kupovat po kusech, ale jako kritické se u tohoto procesoru stalo značné paměťové omezení. Program splňující všechny kladené požadavky potřeboval více Flash paměti. Všechny požadavky uspokojil až použitý procesor SAMD21 mini. Při vývoji na prvních dvou procesor bylo využito projektu mbed, který nabízí uživatelské knihovny pro práci s základními rozhraními pro periferie jako jsou SPI nebo UART. Při práci s posledním procesorem jsem místo toho použil programovací jazyk Arduino, který je stejně jako mbed postavený na C++. Nebylo potřeba psát celý program znovu, pouze bylo potřeba modifikovat již vytvořený software pro mbed.

Případné budoucí modifikace programu jsou značně závislé na použitém hardwaru. Pro práci s každou z periférií jsem vytvořil knihovnu, takže například použití čidla LDC1612, které funguje obdobně jako čidlo LDC1101, ale komunikuje po rozhraní I2C by vyžadovalo pouze výměnu knihovny. Stejně jednoduše nahraditelná je i knihovna pro práci s modulem AX-SFEU.

Literatura

[1] NCS36510 https://www.onsemi.com/pub/Collateral/NCS36510-D.PDF

[2] LPC 82x https://www.nxp.com/docs/en/data-sheet/LPC82X.pdf

[3] SAMD21 http://ww1.microchip.com/downloads/en/DeviceDoc/SAMD21-Family-DataSheet-DS40001882D.pdf

[4] IoT IDK User Guide https://www.onsemi.com/pub/Collateral/AND9666-D.PDF

[5] MCUXpresso https://mcuxpresso.nxp.com/en/welcome

[6] Arduino https://www.arduino.cc

[7] BB-GEVK Board https://www.onsemi.com/PowerSolutions/evalBoard.do?id=BB-GEVK

[8] AX-SFEU https://www.onsemi.com/pub/Collateral/AX-SFEU-D.PDF

[9] LDC1101 http://www.ti.com/lit/ds/symlink/ldc1101.pdf

[10] IoT Technology https://www.arm.com/solutions/iot/iot-technology

[11] Mbed https://www.mbed.com/en/

[12] MbedOS https://os.mbed.com/docs/mbed-os/v5.12/introduction/index.html

[13] Arduino Serial reference https://www.arduino.cc/reference/en/language/functions/communication/serial/

[14] Arduino SPI https://www.arduino.cc/en/reference/SPI

[15] Arduino RTC https://www.arduino.cc/en/Reference/RTC

[16] BARR, Michael. Programming embedded systems in C and C++. Sebastopol, Calif.: O'Reilly, c1999. ISBN 1565923545.